4

Advanced UAV Trajectory Generation: Planning and Guidance

Antonio Barrientos, Pedro Gutiérrez and Julián Colorado
Universidad Politécnica de Madrid – (Robotics and Cybernetics Group)
Spain

1. Introduction

As technology and legislation move forward (JAA & Eurocontrol, 2004) remotely controlled, semi-autonomous or autonomous Unmanned Aerial Systems (UAS) will play a significant role in providing services and enhancing safety and security of the military and civilian community at large (e.g. surveillance and monitoring) (Coifman et al., 2004). The potential market for UAVs is, however, much bigger than just surveillance. UAVs are ideal for risk assessment and neutralization in dangerous areas such as war zones and regions stricken by disaster, including volcanic eruptions, wildfires, floods, and even terrorist acts. As they become more autonomous, UAVs will take on additional roles, such as air-to-air combat and even planetary science exploration (Held et al., 2005).

As the operational capabilities of UAVs are developed there is a perceived need for a significant increase in their level of autonomy, performance, reliability and integration with a controlled airspace full of manned vehicles (military and civilian). As a consequence researchers working with advanced UAVs have moved their focus from system modeling and low-level control to mission planning, supervision and collision avoidance, going from *vehicle* constraints to *mission* constraints (Barrientos et al., 2006). This mission-based approach is most useful for commercial applications where the vehicle must accomplish tasks with a high level of performance and maneuverability. These tasks require flexible and powerful trajectory-generation and guidance capabilities, features lacking in many of the current commercial UAS. For this reason, the purpose of this work is to extend the capabilities of commercially available autopilots for UAVs. Civil systems typically use basic trajectory-generation algorithms, capable only of linear waypoint navigation (Rysdyk, 2003), with a minimum or non-existent control over the trajectory. These systems are highly constrained when maneuverability is a mission requirement. On the other hand, military researchers have developed algorithms for high-performance 3D path planning and obstacle avoidance (Price, 2006), but these are highly proprietary technologies that operate with different mission constraints (target acquisition, threat avoidance and situational awareness) so they cannot be used in civil scenarios.

This chapter presents a robust Trajectory Generation and Guidance Module (TG^2M), a software tool capable of generating complex six-degrees-of-freedom trajectories in 3D space with velocity, acceleration, orientation and time constraints. The TG^2M is an extension module to the Aerial Vehicle Control Language (AVCL), a software architecture and

interpreted language specification that address the issues of mission definition, testing, validation and supervision for UAVs (Barrientos et al., 2006). The AVCL platform incorporates a 3D visual simulator environment that uses a Geographic Information System (GIS) as the world's data model. The GIS backend means all objects are geo-referenced and that several official and commercial databases may be used for mission planning (roads, airports, power lines, crop fields, etc.). The language specification contains a wide set of instructions that allow the off/on-line supervision and the creation of vehicle-independent missions.

The TG²M is designed to model 3D cartesian paths with parametric constraints. The system uses two types of mathematical descriptors for trajectories: analytical functions and polynomial interpolation. The two main contributions of the module are its geometrical representation of the trajectory and its parametric definition. Simple maneuvers like lines and circumference arcs are created with analytical functions that constrain the geometry of the desired path; then the parametric constraints are *applied*. These constraints are typically kinematic: constant velocity and acceleration, trapezoidal curves to accelerate or stop, etc. More complex maneuvers are described with polynomial interpolation and fitted to the critical control path points, meeting desired position, time and velocity constraints. These polynomial functions are based on third and fourth order splines with fixed boundary conditions (for example initial and final velocities), which join all control points with a continuous and smooth path (Jaramillo-Botero et al., 2004).

The section 2 of this chapter presents some of the current techniques extracted from the military aerial survey applications, showing *complex* mission-planning tools capable of addressing the mission-specific constraints required. Within those approaches, this section also introduces a brief description of the AVCL architecture, describing its components, modules, and the main features provisioned, addressing a novel way of human-mission planning definition and testing. The section 3 introduces the AVCL built-in TG²M framework, describing the different techniques used for the trajectory planning and guidance of UAVs, as well as the mathematical treatment of these methods (analytical functions and polynomial interpolation). On the other hand, simulation-based results (see section 4) using a mini-helicopter simulator embedded into the AVCL environment will show the capabilities of the TG²M while flying aggressive and simple maneuvers. Last but not least, "final observations" in section 5 includes comments about the TG²M framework and the upcoming additions to the TG²M under current development.

2. Motion-planning Methodologies

A planning algorithm should provide feasible and flyable optimal trajectories that connect starting with target points, which should be compared and valued using specific criteria. These criteria are generally connected to the following major concerns, which arise during a plan generation procedure: feasibility and optimality. The first concern asks for the production of a plan to safely "move" the UAV to its target state, without taking into account the quality of the produced plan. The second concern asks for the production of optimal, yet feasible, paths, with optimality defined in various ways according to the problem under consideration (LaValle, 2006). Even in simple problems searching for optimality is not a trivial task and in most cases results in excessive computation time, not always available in real-world applications. Therefore, in most cases we search for suboptimal or just feasible solutions. The simplest way to model an UAV path is by using

straight-line segments that connect a number of waypoints, either in 2D or 3D space (Moitra et al., 2003, Zheng et al., 2005). This approach takes into account the fact that in typical UAV missions the shortest paths tend to resemble straight lines that connect waypoints with starting and target points and the vertices of obstacle polygons. Although waypoints can be efficiently used for navigating a flying vehicle, straight-line segments connecting the corresponding waypoints cannot efficiently represent the real path that will be followed by the vehicle due to the own kinematics of the traced path. As a result, these simplified paths cannot be used for an accurate simulation of the movement of the UAV in an optimization procedure, unless a large number of waypoints is adopted. In that case the number of design variables in the optimization procedure explodes, along with the computation time. This is why this section presents some background based on the state-of-the-art mission/path planning for UAVs. The purpose (apart from the state-of-the-art survey) is to compare to the AVCL architecture in order to observe how its TG²M embedded framework (see section 3 for details) is a complement of some lacking approaches found in this specialized literature.

2.1 Background

Researchers at top research centers and universities (e.g. JPL at Caltech, Carnegie Mellon, NASA, ESA, among others) are dealing with the development of new strategies that allow high-level mission UAV planning within the desired flight performance. As examples of these approaches we present the NASA Ames Research Center, the Georgia Institute of Technology, and the University of Illinois.

For the past decade NASA has focused on developing an efficient and robust solution to Obstacle Field Navigation (OFN), allowing a fast planning of smooth and flyable paths around both known and unknown obstacles (Herwitz, 2007). Evolutionary algorithms have been used as a viable candidate to solve path-planning problems effectively, providing feasible solutions within a short time. Given a number of UAVs that are launched from different and known initial locations, the algorithm must create 2-D trajectories with a smooth velocity distribution along each trajectory and with the goal of reaching a predetermined target location, while ensuring collision avoidance and satisfying specific route and coordination constraints and objectives. B-Splines curves are used in order to model both the 2-D trajectories and the velocity distribution along each flight path. A flying laboratory for autonomous systems, based on the Yamaha RMAX helicopter is being developed, which it incorporates the OFN planner and one all-digital camera system with a state-of-the-art tracking and passive ranging capabilities. Machine stereo-vision is being used to determine safe landing areas and monocular vision is used to track the landing location without access to GPS. A Ground Control Station (GCS) is being integrated into the simulation environment to investigate the issues related to high altitude and long endurance flights.

As an alternative approach other researchers do not focus on generating complex trajectories profiles; their aim is to develop robust mission management, capable of successfully integrating the available onboard hardware (e.g. cameras, sensors, etc). The Georgia Tech's UAV Lab developed the GTMax architecture (Alison et al., 2003) based on a Yamaha R-Max mini-helicopter, is an example of this alternative approach. The GTMax system is capable of fully autonomous flight with decentralized software modules for trajectory generation, offline simulation, supervision, guidance and control. The trajectory module generates lineal waypoints between targets with initial and final velocity parameterization. Commands to

the helicopter take the form of different types of waypoints. All trajectories generated are assumed to be physically feasible by the helicopter, the kinematic model used for the trajectory generation uses specifiable limits on the maximum speed and acceleration the aircraft may have during a maneuver. From a high-level mission management perspective the GTMax architecture allows the UAV to autonomously locate targets with an identifying symbol within a designated search area. Once the correct target is identified the mission coordination and flight path generation is done at a centralized location on the ground, and the commands are transmitted to the UAV via a wireless datalink. The GTMax GCS interfaces with the primary flight computer and displays vehicle status, the object tracker information and the flight plans generated by the mission planner. The Vision Monitoring Station receives streaming video from the camera and the output of the image processor. This allows the operator to monitor the efficiency of the image processing as well as visually document the results of the search in the final phase of the mission.

For other projects the main goal is the unification of high-level mission commands and development of generic languages that support online UAV mission planning without constraining the system to a single vehicle. The University of Illinois at Urbana-Champaign (Frazzoli, 2002) presents a new framework for UAV motion planning and coordination. This framework is based on the definition of a modelling language for UAV trajectories, based on the interconnection of a finite number of suitably defined motion primitives, or maneuvers. Once the language is defined, algorithms for the efficient computation of feasible and optimal motion plans are established, allowing the UAV to fulfill the desired path.

If we analyze the UAS described above, almost all of them share one important limitation: their software architecture is tightly coupled to one vehicle and the capabilities of its low-level controller. Civil applications require open and extendable software architectures capable of *talking* to vehicles from different suppliers. The AVCL addresses those limitations, allowing to model different vehicles into a single common language (vehicle-independent missions). In the same fashion the described vehicles show that complex and simple maneuvers could be a suitable solution depending on the kind of mission to fulfill. For this reason the AVCL is extended with the TG²M framework, capable of generating simple and complex 3D paths with the necessary vehicle constrains. The next sub-section introduces the AVCL architecture and some of the features provisioned.

2.2 The Aerial Vehicle Control Language - AVCL

The AVCL is not just a language capable of describing the missions and capabilities of an heterogeneous group of vehicles, it is part of a bigger framework that includes its interpreter, a definition of a base-vehicle, and a Mission Planner that uses GIS as the data-model for the world (Barrientos et al., 2006). The Mission Planner (MP) is not tied to a particular set of vehicles, sensors or commands. At any given time new functionality can be loaded and displayed to the human operator as new options and commands. This means that the MP tool is to be extended through Vehicle and Command Libraries without recompiling, and those new capabilities and better vehicles can be added easily. The Mission Planner is a great tool for simulation and direct comparison of various trajectory trackers, UAV models and controllers, because it can display N missions at the same time.

When considered just as a language the AVCL concept is the abstraction layer that allows the human supervisor to create missions that are vehicle and payload independent, promoting code reuse. At the same time the AVCL statements and commands hide device

specific software/hardware, and serve as mission definition and storage. As an example of the code used to define operations within a mission:

```
uav.Sensors(0) = parser.loadObject ('camera.lib')
uav.Sensors(0).LookAt (p1)
uav.Sensors(1) = parser.loadObject ('laser.lib')
uav.Sensors(1).TurnOn()
uav.doLine (way_points = {p1, p2, p3, p4}, vel = 0.9 m_s)
```

Compared to previous vehicle *programming* languages the AVCL and its interpreter provide several advantages: intuitive handling of different systems of units; its use of the object-oriented-programming paradigm; facilities for inter-vehicle communications; run-time definition of relations between vehicles, sensors and other equipment; it may be extended easily through C, C++ or C# code; the interpreter is a light-weight application written in C++, therefore it may be deployed in many SW/HW architectures. Before the development of the TG²M module the AVCL framework relied on a simpler guidance module to connect waypoints with straight-line segments, and while the language could describe complex maneuvers and mission constraints the framework lacked the capacity to *fly* a vehicle through complex paths.

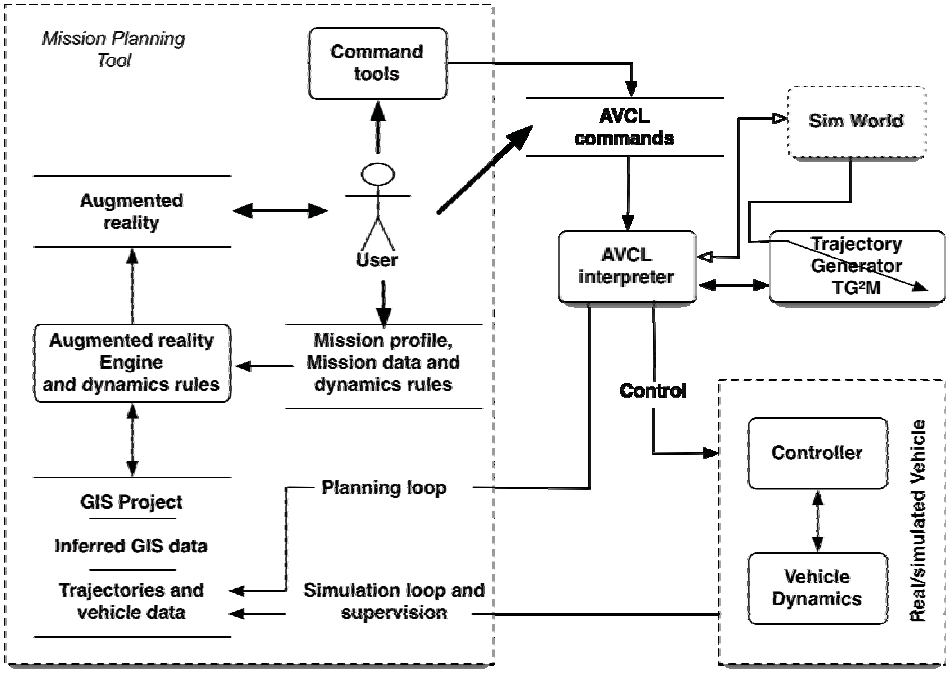

Figure 1. The AVCL simplified diagram for mission planning

3. Trajectory Generation and Guidance Module - TG²M

The TG²M is designed to model 3D cartesian paths with parametric constraints. The system uses two types of mathematical descriptors for trajectories: polynomial interpolation and

analytical functions. Complex maneuvers are described with polynomial interpolation based on third and fourth order splines with fixed boundary conditions (e.g. initial and final velocities user definition), which join all control points with a continuous and smooth path. Likewise, simple maneuvers like lines and circumference are created with analytical functions that constrain the geometry of the desired path. These constraints are typically kinematic: constant velocity and acceleration, trapezoidal curves to accelerate or stop, etc.

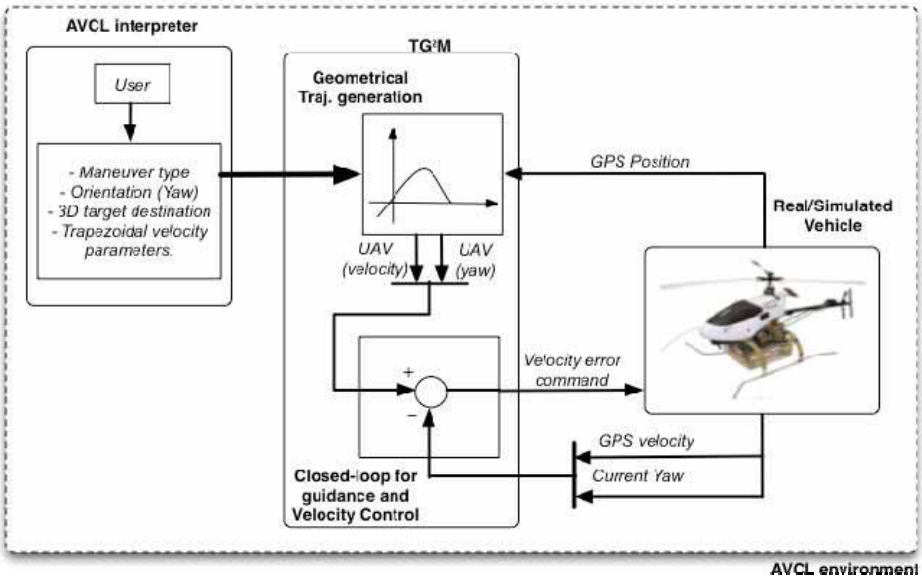

Figure 2. The TG²M framework

As shown in Fig. 2, two main modules compose the TG²M framework: the geometrical trajectory generation and the online closed-loop guidance. All the parameters and fundamental data (e.g. the desired maximum speed, etc) are provided by the user via the AVCL interpreter.

3.1 Polynomial interpolation using splines

The fundamental idea behind the spline interpolation is based on the definition of smooth curves through a number of points. These curves are represented by a polynomial function (normally of third-grade) defined by some coefficients that determine the spline used to interpolate the numerical data points. These coefficients *bend* the line so that it passes through each of the data points without any erratic behavior or breaks in continuity. The essential idea is to define a third-degree polynomial function $S(t)$ of the form:

$$S(t) = a_i t^3 + b_i t^2 + c_i t + d_i \tag{1}$$

This third-degree polynomial needs to conform to the following conditions in order to interpolate the desired knot-points as depicted in Fig. 3:

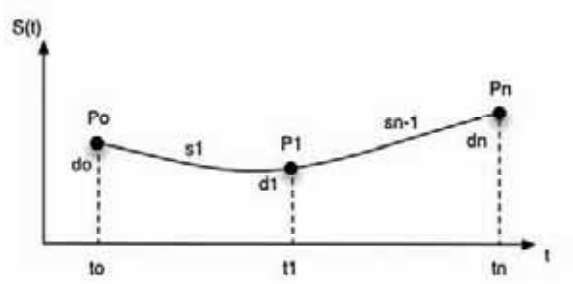

Figure 3. Knot-points to interpolate using 3D splines (with free boundary conditions)

- $S(t)$, $S'(t)$ and $S''(t)$ will be continuous on the interval $[t_0, t_n]$.
- Since the curve $S(t)$ must be continuous across its entire interval, it can be concluded that each sub-function must joint at the data points, so: $s_i(t_i) = s_{i-1}(t_i)$.

Taking into account the set of conditions previously described, for each $i = 1, 2.....n-1$, $t_i \in [t_i, t_{i+1}]$ $S(t_i) = s_i(t_i)$, and letting $h_i = \sum_{i=1}^{n-1} t_{i+1} - t_i$, Eq. (1) is re-writing as:

$$s_i(t_i) = a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i \tag{2}$$

Also, to make the curve smooth across the interval, the first derivative of the $S(t)$ function must be equal to the derivative of the position reference points; this yields:

$$\begin{aligned} s_i'(t_i) &= 3a_i h_i^2 + 2b_i h_i + c_i \\ s_i'(t_i) &= s_{i-1}'(t_i) \\ s_i'(t_i) &= c_i = 3a_{i-1} h_i^2 + 2b_{i-1} h_i + c_{i-1} \end{aligned} \tag{3}$$

Applying the same approach for the second derivative:

$$\begin{aligned} s_i''(t_i) &= 6a_i h_i + 2b_i \\ s_i''(t_i) &= s_{i+1}''(t_i) \\ s_i''(t_i) &= 2b_{i+1} = 6a_i h_i + 2b_i \end{aligned} \tag{4}$$

For the solution of the polynomial coefficients in Eq. (1), the system in Eq. (5) must be solved as:

$$A \cdot Y = f \tag{5}$$

where the matrix $A \in \Re^{nxn}$ (n is the number of knot-points to interpolate) corresponds to:

$$A=\begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \ddots & & & \vdots \\ 0 & h_0 & 2(h_1+h_2) & h_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

The term $h \in \Re^{n-1}$ in Eq. (6) is the time vector defined for each point $(P_0, P_1, ...P_n)$. The b_i coefficients in Eq. (1) are stacked in $Y \in \Re^{nx3}$, which yields the term $f \in \Re^{nx3}$ as:

$$f=\begin{bmatrix} 0 \\ \frac{3}{h_1}(d_2-d_1)-\frac{3}{h_0}(d_1-d_0) \\ \vdots \\ \frac{3}{h_{n-1}}(d_n-d_{n-1})-\frac{3}{h_{n-2}}(d_{n-1}-d_{n-2}) \\ 0 \end{bmatrix} \tag{7}$$

From Eq. (2) and (3), the a_i and c_i coefficients are respectively obtained as:

$$\begin{aligned} a_i &= \frac{b_{i+1}-b_i}{3h_i} \\ c_i &= \frac{1}{h_i}(d_{i+1}-d_i)-\frac{h_i}{3}(2b_i-b_{i+1}) \\ d_i &= S(t_i) \end{aligned} \tag{8}$$

Example 1. *Simple UAV trajectory planning using 3D splines with fixed boundary condition.*

Let's define three knot-points as: $P_0=[0,0,0]$, $P_1=[5,10,10]$, $P_2=[0,10,20]$ *with the following time condition for each point:* $t=[0,10,20]$*(note that the time vector components are given in seconds). Calculate a 3D spline considering zero initial and final velocities.*

The natural 3D splines with free boundary conditions may generate smooth paths but without control over the velocities over the knot-points. Because of this, Eq. (6) and (7) must be complemented in order to generate a 3D spline with the fixed boundary conditions, in this case, with zero initial and final velocities. Re-writing those equations yield:

$$A=\begin{bmatrix} 2h_0 & h_0 & 0 & \cdots & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \ddots & & & \vdots \\ 0 & h_0 & 2(h_1+h_2) & h_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix} \tag{9}$$

The same system $A \cdot Y = f$ must be solved with the new $A \in \Re^{nxn}$ from Eq. (9) and $f \in \Re^{nx3}$ defined in Eq. (10). Note that the first derivative of the $S(t)$ function has been added in the first and last position of the f vector, allowing the control of the initial and final velocities of the curve.

$$f = \begin{bmatrix} \frac{3}{h_0}(d_1 - d_0) - 3S'(t_0) \\ \frac{3}{h_1}(d_2 - d_1) - \frac{3}{h_0}(d_1 - d_0) \\ \vdots \\ \frac{3}{h_{n-1}}(d_n - d_{n-1}) - \frac{3}{h_{n-2}}(d_{n-1} - d_{n-2}) \\ 3S'(t_n) - \frac{3}{h_{n-1}}(d_n - d_{n-1}) \end{bmatrix} \tag{10}$$

The first procedure is to obtain the time vector as follows: $h = \sum_{i=1}^{n-1} t_{i+1} - t_i = \begin{bmatrix} 10 & 10 \end{bmatrix}$, then the system: $Y = A^{-1} f$ must be solved to obtain the polynomial coefficients as:

$$A = \begin{bmatrix} 20 & 10 & 0 \\ 10 & 40 & 10 \\ 0 & 10 & 20 \end{bmatrix}, \quad Y_x = \begin{bmatrix} b_{0x} = 0.15 \\ b_{1x} = -0.15 \\ b_{2x} = 0.15 \end{bmatrix}, \quad Y_y = \begin{bmatrix} b_{0y} = 0.15 \\ b_{1y} = 0 \\ b_{2y} = -0.15 \end{bmatrix}, \quad Y_z = \begin{bmatrix} b_{0z} = 0.1125 \\ b_{1z} = 0.0750 \\ b_{2z} = -0.2625 \end{bmatrix}$$

$$f_x = \begin{bmatrix} 1.5 \\ -3 \\ 1.5 \end{bmatrix} \quad f_y = \begin{bmatrix} 3 \\ 0 \\ -3 \end{bmatrix} \quad f_z = \begin{bmatrix} 3 \\ 1.5 \\ -4.5 \end{bmatrix}$$

Finally, the two polynomials for each (x, y, z) component are evaluated from the time [0, 10] to [10, 20]. Figure 4 shows the results.

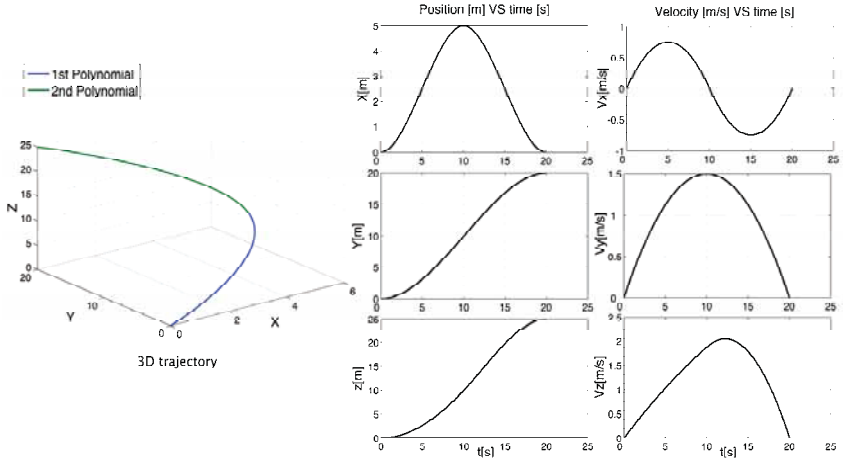

Figure 4. 3D spline with fixed boundary condition

The end of Example 1: Simple UAV trajectory planning using 3D splines

In the previous example, the 3D splines with fixed boundary conditions allowed to define a smooth curve across the knot-points. Nevertheless, two basic problems must be taken into account: the 3D spline only allows the user to establish the initial and the final velocities of the whole trajectory, limiting the user to have total control over the other points. The second problem is the smoothness of the acceleration curves (linear). To solve these problems 4D splines must be used. These address the user total control over the velocity profile across the whole trajectory and its results in additional *smoothness* for position, velocity and even acceleration curves. This could be more effective for UAVs tasks where the mission requires a strong control of the vehicle acceleration and velocities at each defined knot-point.

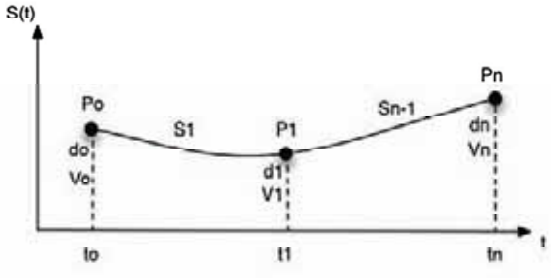

Figure 5. Knot-points to interpolate with 4D splines

The fourth-degree polynomial is defined by:

$$S(t) = e_i t^4 + a_i t^3 + b_i t^2 + c_i t + d_i \tag{11}$$

The same 3D-spline conditions previously described also apply to the 4D-spline. To obtain a generalized solution of the system to solve ($A \cdot Y = f$), we start from the three-point case as depicted in Fig. 5. The polynomials for each trajectory segment as a function of time *t* are:

$$\begin{aligned}
s_n(t_0) &= e_0 t_0^4 + a_0 t_0^3 + b_0 t_0^2 + c_0 t_0 + d_0 = f(t_0) \\
s_n(t_1) &= e_0 t_1^4 + a_0 t_1^3 + b_0 t_1^2 + c_0 t_1 + d_0 = f(t_1) \\
s_{n-1}(t_1) &= e_1 t_1^4 + a_1 t_1^3 + b_1 t_1^2 + c_1 t_1 + d_1 = f(t_1) \\
s_{n-1}(t_n) &= e_1 t_n^4 + a_1 t_n^3 + b_1 t_n^2 + c_1 t_n + d_1 = f(t_n)
\end{aligned} \tag{12}$$

Taking the first and second derivatives (velocities and accelerations), we obtain:

$$\begin{aligned}
s_1'(t_0) &= 4e_0 t_0^3 + 3a_0 t_0^2 + 2b_0 t_0 + c_0 = V_0 \\
s_1'(t_1) &= 4e_0 t_1^3 + 3a_0 t_1^2 + 2b_0 t_1 + c_0 = V_1 \\
s_{n-1}'(t_1) &= 4e_1 t_1^3 + 3a_1 t_1^2 + 2b_1 t_1 + c_1 = V_1 \\
s_{n-1}'(t_n) &= 4e_1 t_n^3 + 3a_1 t_n^2 + 2b_1 t_n + c_1 = V_n
\end{aligned} \tag{13}$$

The second derivatives of Eq. (12) yield a set of accelerations. Equaling the acceleration functions for the intermediate points (t_1 for each case) and setting to zero the initial and final acceleration of the path segment yields:

$$\begin{aligned} s_0''(t_0) &= 12e_0t_0^2 + 6a_0t_0 + 2b_0 = s_{n-1}'(t_1) = 12e_1t_1^2 + 6a_1t_1 + 2b_1 \\ s_0''(t_1) &= 12e_0t_1^3 + 6a_0t_1 + 2b_0 = s_{n-1}''(t_n) = 12e_1t_n^2 + 6a_1t_n + 2b_1 \end{aligned} \tag{14}$$

Equations (12), (13) and (14) conform the complete system in order to obtain the ten polynomial coefficients. Solving $A \cdot Y = f$, the matrix $A \in \Re^{5(n-1)\,x\,5(n-1)}$ corresponds to:

$$A = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & 0 & 0 & 0 & 0 & 0 \\ 1 & t_1 & t_1^2 & t_1^3 & t_1^4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & t_1 & t_1^2 & t_1^3 & t_1^4 \\ 0 & 0 & 0 & 0 & 0 & 1 & t_2 & t_2^2 & t_2^3 & t_2^4 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2t_1 & 3t_1^2 & 4t_1^3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2t_1 & 3t_1^2 & 4t_1^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2t_2 & 3t_2^2 & 4t_2^3 \\ 0 & 0 & 2 & 6t_1 & 12t_1^2 & 0 & 0 & -2 & -6t_1 & -12t_1^2 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 0 & 0 & -2 & -6t_2 & -12t_2^2 \end{bmatrix} \tag{15}$$

The polynomial coefficients are stacked into the $Y \in \Re^{5(n-1)}$ vector, and with the $f \in \Re^{5(n-1)}$ term, the total system is defined from Eq. (14) and (15):

$$\begin{aligned} Y &= \begin{bmatrix} d_0 & c_0 & b_0 & a_0 & e_0 & d_1 & c_1 & b_1 & a_1 & e_1 \end{bmatrix}^T \\ f &= \begin{bmatrix} f(t_0) & f(t_1) & f(t_1) & f(t_n) & V_0 & V_1 & V_1 & V_n & 0 & 0 \end{bmatrix}^T \end{aligned} \tag{16}$$

Example 2. *UAV trajectory planning using 4D splines with total velocity control*

Lets define three knot-points as: $P_0 = [0,0,0]$, $P_1 = [5,7,5]$, $P_2 = [4,2,10]$ *in the time* $t = [0,4,8]$ *(note that the time vector components are given in seconds). Calculate the 4D spline considering the following velocity profile:* $V_0 = [0,0,0]$, $V_1 = [0.5,0.8,1]$, $V_2 = [1,1.5,1]$.

Solving the system: $Y = A^{-1}f$, we obtain the following numerical data:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 16 & 64 & 256 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 16 & 64 & 256 \\ 0 & 0 & 0 & 0 & 0 & 1 & 8 & 64 & 512 & 4096 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 8 & 48 & 256 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 8 & 48 & 256 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 16 & 192 & 2048 \\ 0 & 0 & 2 & 24 & 192 & 0 & 0 & -2 & -24 & -192 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \end{bmatrix}, \quad Y_{x,y,z} = \begin{bmatrix} d_{0x} & d_{0y} & d_{0z} \\ c_{0x} & c_{0y} & c_{0z} \\ b_{0x} & b_{0y} & b_{0z} \\ a_{0x} & a_{0y} & a_{0z} \\ e_{0x} & e_{0y} & e_{0z} \\ d_{1x} & d_{1y} & d_{1z} \\ c_{1x} & c_{1y} & c_{1z} \\ b_{1x} & b_{1y} & b_{1z} \\ a_{1x} & a_{1y} & a_{1z} \\ e_{1x} & e_{1y} & e_{1z} \end{bmatrix}, \quad f_{x,y,z} = \begin{bmatrix} 0 & 0 & 0 \\ 5 & 7 & 5 \\ 5 & 7 & 5 \\ 4 & 2 & 10 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad f_{x,y,z} = \begin{bmatrix} 0 & 0 & 0 \\ 5 & 7 & 5 \\ 5 & 7 & 5 \\ 4 & 2 & 10 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Finally, the two polynomials for each (x, y, z) component are evaluated from the time [0, 4] to [4, 8] seconds. Figure 6 shows the results:

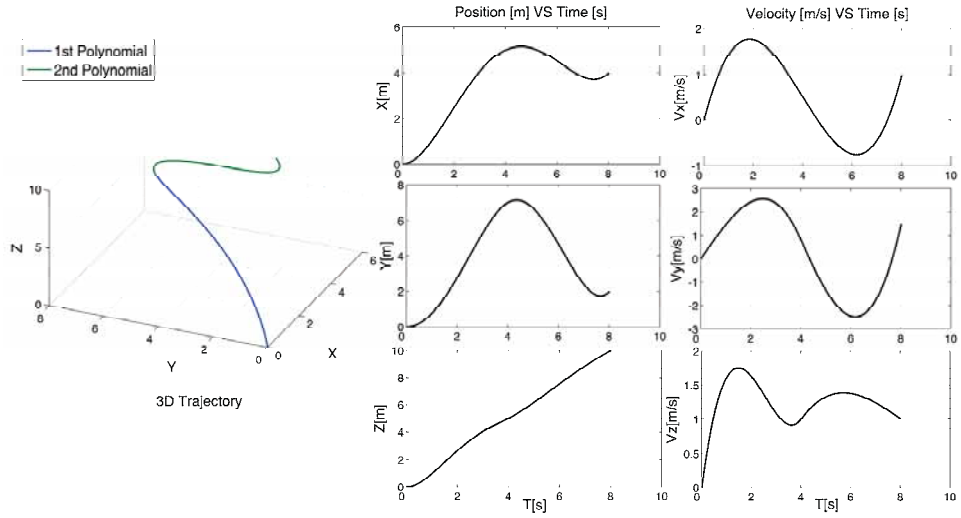

Figure 6. 4D spline with total control of the knot-points velocities

The end of Example 2: UAV trajectory planning using 4D splines with total velocity control

3.2 Simple Maneuvers with Analytical Functions

For simple maneuvers the TG²M framework also supports the definition of straight-lines and circumferences via analytical functions that constrain the geometry of the desired path. These constraints are typically kinematic: constant velocity and acceleration, trapezoidal curves to accelerate or stop, etc. This kind of parameterization is useful when the mission requires the UAV stops at the desired end-point of the trajectory effectively, due to the user-control of the acceleration slope tilt level. For both (lines and circumferences) the desired set of velocities must fulfill the following trapezoidal velocity profile:

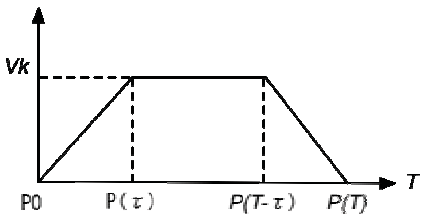

Figure 7. Trapezoidal velocity profile used for simple UAV maneuvers

From Fig. 7, three fundamental segments compose the total function to define the straight-line. The two intermediate points of the trapezoidal curve: $P(\tau)$ $P(T-\tau)$ are calculated as [1]:

[1] Sub-indices *x,y,z* refer to each coordinate of motion.

$$P(\tau)_{x,y,z} = \frac{1}{2\left\|P(T)_{x,y,z} - P_{0x,y,z}\right\|}\frac{V_k}{t}t^2\left(P(T)_{x,y,z} - P_{0x,y,z}\right) + P_{0x,y,z}$$

$$P(T-\tau)_{x,y,z} = \frac{V_k(T-2t)}{\left\|P(T)_{x,y,z} - P_{0x,y,z}\right\|}\left(P(T)_{x,y,z} - P_{0x,y,z}\right) + P(\tau)_{x,y,z} \quad (17)$$

Once that the intermediate points have been defined, the three segments that compose the total function are defined by Eq. (18). In the first section (from P_0 to $P(\tau)$) the initial velocity is set $V_i = 0$ and progresses toward a final velocity V_k. The second segment (from $P(\tau)$ to $P(T-\tau)$) is traced at constant maximum velocity V_k. Finally the last segment (from $P(T-\tau)$to $P(T)$) drives the vehicle from V_k to zero velocity.

$$f(t) = \begin{cases} \dfrac{V_k}{2\tau\left(P(\tau) - P_0\right.}t^2\left(P(\tau) - P_0\right) + P_0 & 0 \le t \le \tau \\ \dfrac{V_k}{\left(P(T-\tau) - P(\tau)\right.}(t-\tau)\left(P(T-\tau) - P(\tau)\right) + P(\tau) & \tau < t \le T-\tau \\ \dfrac{(t-T+\tau)}{\left(P(T) - P(T-\tau)\right.}\left(-\dfrac{V_k}{2\tau}(t-T+\tau) + V_k\right)\left(P(T) - P(T-\tau)\right) + P(T-\tau) & T-\tau < t \quad T \le \end{cases} \quad (18)$$

The same approach is applied to generate circumferences with the trapezoidal profile used for the straight-lines. Three knot-points define the circumference path and the objective is to find the trace angle across the trajectory.

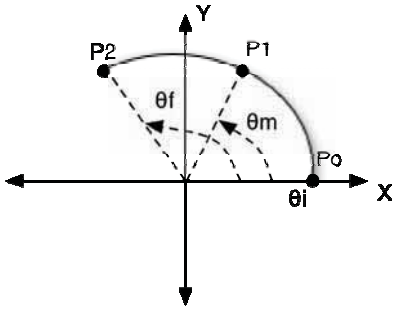

Figure 8. X-Y plane for circumference maneuver

The first step is to determine the center of the circle that joins the three knot-points. The equation of the trajectory plane is defined by the cross product between the vectors formed by points P_0, P_1 and P_2, as shown in Eq. (19).

$$P_0\vec{P}_1 \times P_0\vec{P}_2 = N(A,B,C)$$
$$Ax + By + Cz = cte \quad (19)$$

The center of the circle $c\left(x_c, y_c, z_c\right)$ is always equidistant to any point over the circle[2], then:

$$\begin{aligned} &\left(x_{0,1,2} - x_c\right)^2 + \left(y_{0,1,2} - y_c\right)^2 + \left(z_{0,1,2} - z_c\right)^2 = K^2 \\ &Ax_c + By_c + Cz_c = cte \end{aligned} \tag{20}$$

To obtain the relation between the angle motion (θ), the angular velocity (ω) and the tangential velocity (v_t), constant velocity is assumed across the path, yielding:

$$\begin{aligned} &\omega = \frac{d\theta}{dt} = \frac{v_t}{r} \\ &d\theta = \frac{v_t}{r} dt \\ &\int d\theta = \int \frac{v_t}{r} dt \\ &\theta\left(t\right) = \frac{v_t}{r} t + \theta_i \quad \text{where} \quad t \in [0, T] \end{aligned} \tag{21}$$

Likewise, the relation between the angle motion (θ) and the angular acceleration of the curve is given by:

$$\begin{aligned} &at + v_0 = r\frac{d\theta}{dt} \\ &d\theta = \frac{at + v_0}{r} dt \\ &\int d\theta = \int \frac{at + v_0}{r} dt \\ &\theta\left(t\right) = \frac{a}{2r} t^2 + \frac{v_0}{r} t + \theta_i \quad \text{where} \quad t \in [0, T] \end{aligned} \tag{22}$$

If the known parameter is the total motion time (T) of the trajectory, we set the acceleration term to the left-side of the equation, yielding:

$$\begin{aligned} &\theta\left(t\right) = \theta_f = \frac{a}{2r} t^2 + \frac{v_0}{r} t + \theta_i \\ &a = \frac{2\left[\left(\theta_f - \theta_i\right) - v_0 T\right]}{T^2} \end{aligned} \tag{23}$$

Otherwise, if the constrained parameters are the initial and final velocity, the acceleration function is given by:

$$\begin{aligned} &a = \frac{v_{ft} - v_{0t}}{\mathrm{T}} \\ &T = \frac{2\left(\theta_f - \theta_i\right)}{v_{0t} + v_{ft}} \end{aligned} \tag{24}$$

[2] Sub-indices *0,1,2* refers to each vector component for Po,P1,P2 (see Fig. 8).

Example 3. UAV straight-line trajectory with trapezoidal velocity profile

Lets defined a straight-line trajectory starting from $P_0 = [0,0,20]$ *to* $P(T) = [10,0,20]$ *with 10 meters of displacement in the X coordinate at a constant altitude of 20 meters. Define the 3D cartesian trajectory with a maximum velocity of 1m/s taking into account a trapezoidal velocity profile with 30% of acceleration time.*

The distance vector to trace is: $dist = P(T) - P_0 = [10,0,0]$.

The total velocity is obtained from the norm of the vector *dist:* $V_T = \|dist\| = 10$.

The total motion time is calculated as: $T = \dfrac{V_T}{V_k(1-pt)} = 7$, where V_k is the maximum velocity at 1m/s and *pt* is the percentage of acceleration (30%=0.3).

The acceleration motion is also obtained as: $a = \dfrac{V_k}{T * pt} = 0.2381$.

The number of the total points to generate is: $n = 28$ (the user defines this parameter depending on the UAV controller data rate). In the same way, the number of points in the first and last segment of the trapezoidal profile is calculated as: $k = 2(pt \cdot n) = 16$, and for the constant velocity segment: $k_c = n - 2k = 12$. Once that the preliminary data have been calculated, Eq. (16) is used to obtain the intermediate points (see Fig. 7):

$$Px(\tau) = \frac{1}{2V_t} at^2 (dist)_x + P_{0x} = 2.10 \qquad Px(T-\tau) = \frac{V_k(T-2t)}{V_T}(dist)_x + P(\tau)_x = 7.7$$

$$Py(\tau) = \frac{1}{2V_t} at^2 (dist)_y + P_{0y} = 0 \qquad Py(T-\tau) = \frac{V_k(T-2t)}{V_T}(dist)_y + P(\tau)_y = 0$$

$$Pz(\tau) = \frac{1}{2V_t} at^2 (dist)_z + P_{0z} = 20 \qquad Pz(T-\tau) = \frac{V_k(T-2t)}{V_T}(dist)_z + P(\tau)_z = 20$$

The first segment of the path (from P_0 to $P(\tau)$) is calculated using Eq. (17):

$$fx(t) = \sum_{i=1}^{k/2} \frac{V_k}{2t\|P(t) - P_0\|_x} t_i^2 (P(t)_x - P_{0x}) + P_{0x}$$

$$fy(t) = \sum_{i=1}^{k/2} \frac{V_k}{2t\|P(t) - P_0\|_y} t_i^2 (P(t)_y - P_{0y}) + P_{0y} \quad 0 \le t \le \tau$$

$$fz(t) = \sum_{i=1}^{k/2} \frac{V_k}{2t\|P(t) - P_0\|_y} t_i^2 (P(t)_z - P_{0z}) + P_{0z}$$

The time vector t is obtained as: $t = \left[0 : \dfrac{1}{V_k}\dfrac{2\|P(\tau) - P_0\|}{k} : \dfrac{1}{V_k} 2\|P(\tau) - P_0\|\right]$, which is:

$t = [0 \quad 0.5250 \quad 1.0500 \quad 1.5750 \quad 2.1000 \quad 2.6250 \quad 3.1500 \quad 3.6750 \quad 4.2000]$.

For the second segment (at constant velocity V_k), the time vector is calculated as:

$$t = \left[0 : \frac{1}{n-2k} : 1\right] == \left[0 \;\; 0.0833 \;\; 0.1667 \;\; 0.25 \;\; 0.3333 \;\; 0.4167 \;\; 0.5 \;\; 0.5833 \;\; 0.6667 \;\; 0.75 \;\; 0.83 \;\; 0.9167\right]$$

$$fx(t) = \sum_{i-1}^{kc} t_i \left(P(T-\tau)_x - P(\tau)_x\right) + P(\tau)_x$$

$$fy(t) = \sum_{i-1}^{kc} t_i \left(P(T-\tau)_y - P(\tau)_y\right) + P(\tau)_y \qquad \tau < t \le T-\tau$$

$$fz(t) = \sum_{i-1}^{kc} t_i \left(P(T-\tau)_z - P(\tau)_z\right) + P(\tau)_z$$

Finally the last segment drives the vehicle from V_k to zero end velocity. The time vector is:

$$t = \left[0 : \frac{1}{V_k} \frac{2\left\|P(T-\tau) - P(\tau)\right\|}{k} : \frac{1}{V_k} 2\left\|P(T-\tau) - P(\tau)_0\right\|\right] = \begin{bmatrix} 0 \;\; 0.287 \;\; 0.575 \;\; 0.862 \;\; 1.15 \;\; 1.43 \\ 1.72 \;\; 2.01 \;\; 2.3 \end{bmatrix},$$

yielding the following function for each coordinate of motion:

$$f(t)_x = \sum_{i=1}^{k/2} \frac{-V_k t_i^2 + V_k t_i}{2\left\|P(T) - P(T-t)\right\|_x} \left(P(T)_x - P(T-t)_x\right) + P(T-t)_x$$

$$f(t)_y = \sum_{i=1}^{k/2} \frac{-V_k t_i^2 + V_k t_i}{2\left\|P(T) - P(T-t)\right\|_y} \left(P(T)_y - P(T-t)_y\right) + P(T-t)_y \qquad T-t < t \le T$$

$$f(t)_z = \sum_{i=1}^{k/2} \frac{-V_k t_i^2 + V_k t_i}{2\left\|P(T) - P(T-t)\right\|_z} \left(P(T)_z - P(T-t)_z\right) + P(T-t)_z$$

The results are presented in Fig. 9:

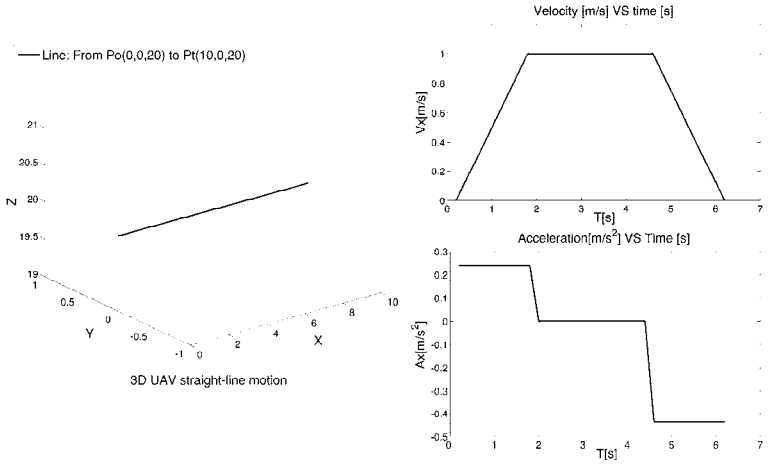

Figure 9. UAV straight-line motion with trapezoidal velocity profile

The end of Example 3: UAV straight-line trajectory with trapezoidal velocity profile

3.3 UAV Guidance

The geometrical representation of a UAV trajectory has been presented in the previous sub-section. Complex trajectories may be described using third-fourth order degree splines, or simple maneuvers may be generated using common lines and circumferences functions with some parametric features defined by the end-user. But in order to complete the generation of trajectories for a single UAV a guidance module is required.

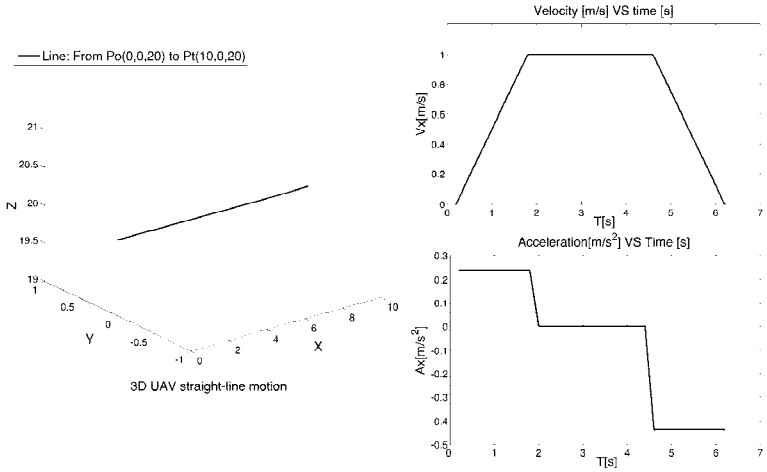

Figure 10. TG²M guidance scheme

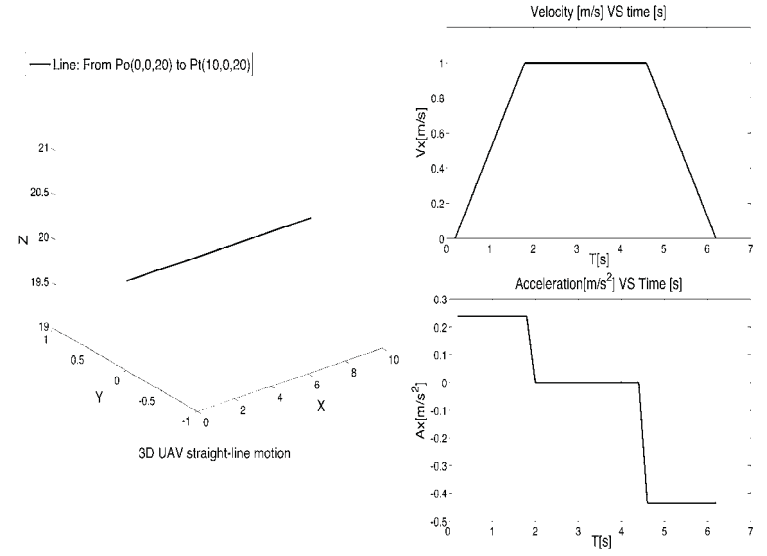

Figure 11. Velocity error command

The AVCL simulation module contains a dynamic model of a mini-helicopter (Valero, 2005) and its associated controllers: attitude, velocity and position. The velocity controller is based

on a Proportional-Integrative "PI" design; it receives the *velocity error command* generated by the guidance module shown in Fig. 10. These velocity error references are named *Vxref* and *Vyref*, with respect to either the world or vehicle's frame, and are calculated with the UAV's error position. In Fig. 11 the theoretical (or ideal) UAV velocity vector is represented by the *Vt* term. Due to wind and other perturbations during flight, a velocity error vector *Ve* must be considered in order to set the final velocity references to send to the vehicle's controller. This vector is derived from the UAV position error between the current and desired positions. Finally, the velocity references *Vxref* and *Vyref* are the components of the vector *Vt* +*Ve*.

The guidance module must take into account that the helicopter's orientation (yaw) is *independent* from the direction of travel. The high-level modules must generate a set of orientations across the vehicle's trajectory. The built-in AVCL control module (see Fig. 10) is capable of receiving *velocity* and *yaw* angle orientation commands from the TG²M module and generating the needed commands for the attitude controller that governs the vehicle's roll and pitch. The TG²M's guidance module focuses on the generation of *yaw* references, and use a simple Proportional "P" controller for smooth *yaw* angle transition during the flight. Two methods are use to generate the *yaw* angle references: for simple maneuvers the *yaw* angle is calculated using simple trigonometric relations due to the path displacement. For complex trajectories using splines we introduce a feasible method to calculate a smooth *yaw* angle evolution. This method also shows how to calculated roll and pitch references due to the vehicle trajectory and its velocity. For *roll, pitch* and *yaw* angles calculation, the following frame of reference is used:

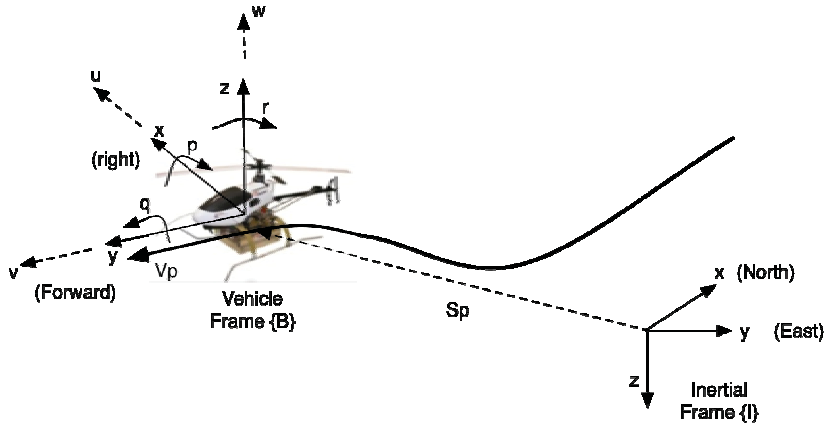

Figure 12. Frames of reference for UAV guidance and control

From Fig. 12, the following vectors can describe the motion of the UAV vehicle in 6 DOF:

$$\eta = \left[\eta_1, \eta_2\right]^T = \left[x, y, z, \phi, \theta, \psi\right]^T$$
$$v = \left[v_1, v_2\right]^T = \left[u, v, w, p, q, r\right]^T \quad (25)$$

In Eq. (25), η_1 denotes the position of the center of mass CM of the vehicle and η_2 its orientation described by the *Euler angles* with respect to the inertial frame {I}. The vector v_1 refers to the linear velocity and v_2 to the angular velocity of vehicle frame {B} with respect to inertial frame {I}. In order to express the velocity quantities between both frames of references (from {B} to {I} and vice versa), the following transformation matrix is used[3]:

$$\begin{aligned} \eta_1' &= R_B^I v_1 \\ \eta_1' &= \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\theta s\phi \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi \quad s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} v_1 \end{aligned} \tag{26}$$

The body-fixed angular velocities and the rate of the Euler angles are related through:

$$\eta_2' = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi / c\theta & c\phi / c\theta \end{bmatrix} v_2 \tag{27}$$

The position and the magnitude of the velocity vector at a point *P* on the trajectory are given by [4]:

$$\begin{aligned} S_p &= \left[x, y, z\right]^T \\ V_p &= \left\| S_p' \right\| = \sqrt{x'^2 + y'^2 + z'^2} \end{aligned} \tag{28}$$

The method to define the Euler angles is based on the Frenet-Serret theory (Angeles, 1997). To every point of the curve we can associate an orthonormal triad of vectors (a set of unit vectors that are mutually orthogonal) namely the tangent e_t, the normal e_n and the bio-normal e_b (see Fig. 13). The Frenet-Serret theory says that by properly arranging these vectors in a matrix $\in \Re^{3x3}$, we obtain a description of the curve orientation due to the position, velocity and acceleration of the UAV while tracing out the path. The unit vectors are then defined as:

$$e_t = \frac{S_p'}{V_p}, \quad e_b = \frac{\left(S_p' \times S_p''\right)}{\left\| S_p' \times S_p'' \right\|}, \quad e_n = e_b \times e_t \tag{29}$$

In the definition of a frame associated with the curve the original definition of the Frenet frame for counterclockwise rotating curves is used; in the case of a clockwise rotating curve, the z axis of the Frenet frame points in the opposite direction upwards than the inertial {I} frame. So in order to define small relative rotation angles for the orientation of a vehicle rotating clockwise and having its z_b axis pointing downwards, we define a reference frame

[3] where $c = \cos()$; $s = \sin()$; $t = \tan()$

[4] The terms (x,y,z) are computed due to the spline methodology previously exposed.

associated with the curve as previously, but rotated with respect to the Frenet by an angle of 180 degrees about the *x*–axis of the Frenet frame (see Fig. 13).

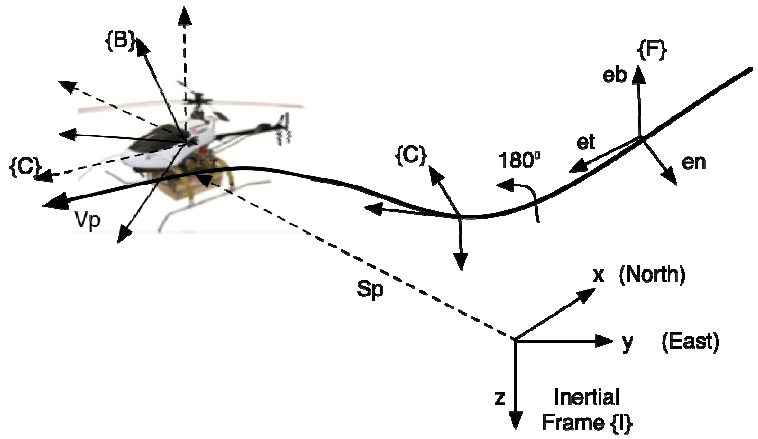

Figure 13. The inertial, the Frenet, the vehicle and the curve Frames

Collectively we denote the Frenet and the rotated frame as the "curve" frame {C}. According to the notation of rotational transformations used in robotics literature, we can express the coordinates of a vector given in the curve frame {C} to the {I} frame with the matrix:

$$\begin{aligned} R_C^I &= \begin{bmatrix} e_t & e_n & e_b \end{bmatrix} \\ R_I^C &= R_C^{I\,T} \end{aligned} \tag{30}$$

For a counterclockwise rotation: $R_C^I = R_x\left(180^\circ\right)\begin{bmatrix} e_t & e_n & e_b \end{bmatrix}$. Likewise, the rotation of the {B} frame from the {C} frame to the reference {R} frame can be expressed using customary aeronautical notation by considering the sideslip angle β and angle of attack α, (Siouris, 2004):

$$\begin{aligned} \beta &= \sin^{-1}\left(\frac{v_R}{V_p}\right) \\ \alpha &= \tan^{-1}\left(\frac{w_R}{u_R}\right) \end{aligned} \tag{31}$$

The vector v_R refers to the *y*-axis velocity component in the reference frame and w_R, u_R to the *z* and *x* - axis respectively. The overall rotation is composed by a rotation about body z_B axis through the angle β, followed by a rotation about the body y_B through the angle α, which is expressed as:

$$R_C^R = R_y^T\left(\alpha\right)R_z^T\left(-\beta\right) \tag{32}$$

Finally, the *roll, pitch* and *yaw* angles can be deduced as follows:

$$\begin{aligned} R_I^R &= R_C^R R_I^C \\ \phi &= \text{atan2}\left(r_{23}, r_{33}\right) \\ \theta &= \text{atan2}\left(-r_{13}, \sqrt{r_{23}^2 + r_{33}^2}\right) \\ \psi &= \text{atan2}\left(r_{12}, r_{11}\right) \end{aligned} \tag{33}$$

Where $r_{i,j}$ represent the components of the rotation matrix $R_I^R \in \Re^{3x3}$. Computing the previous methodology, we use 3D splines with fixed boundary conditions in order to generate a complicated path as shown in Fig. 11. Seven knot-points have been used (distance are in meters): P=[0 0 0; 5 1 2; 10 5 5; 15 10 10; 10 15 15; 5 10 20; 0 8 20; 5 0 20]. Eq. (33) has been used to obtain the UAV orientation with respect to the Inertial Frame as:

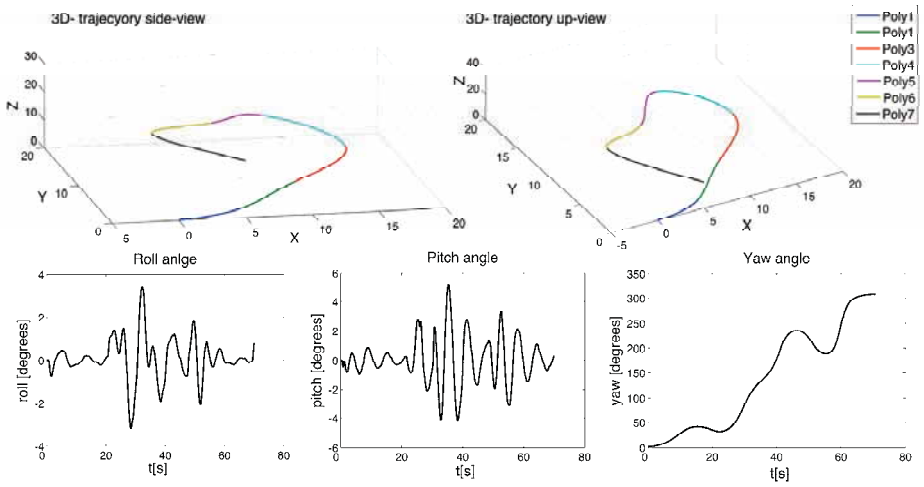

Figure 14. Roll, Pitch and Yaw angle references for UAV guidance

For complex maneuvers the Frenet theory allowed to define smooth *yaw* references as well as *roll* and *pitch* angles if it is required. Nevertheless, the computational cost of calculating those equations could decrease the system performance if the number of knot-points of the path is large. For this reason, normal maneuvers such as straight-lines use simple trigonometric theory to obtain the UAV orientation. On the other hand, the end-user is able to define the kind of orientation of the vehicle, this means that the vehicle is not constrained to be oriented just by the trajectory direction, hence, it will be able to trace out the trajectory oriented towards to any fixed point defined.

The *yaw* angle defined by the term ψ is given by:

$$\psi = \tan^{-1}\left(\frac{y_{diff}}{x_{diff}}\right) \tag{34}$$

Where x_{diff}, y_{diff} correspond to the difference between the target fixed point and the current position of the UAV. In addition, depending of the motion quadrant, the term ψ in Eq. (34) must be fixed, this means that for the $\{x^-,y^+\}$ and the $\{x^-,y^-\}$ quadrant, the *yaw* angle is $\psi = \psi + \pi$, otherwise, for the $\{x^+,y^-\}$ quadrant, $\psi = \psi + 2\pi$.
Figure 15 shows a circumference-arc generated using Eq. (24) as well as the *yaw* evolution of the UAV oriented towards the center of the arc located at [0,0] coordinate in the *x-y* plane using the previosly theory described in Eq. (34).
This section has successfully introduced the mathematical treatment and methods for the generation of complex trajectories and simple maneuvers using the available theory reported in specialized literature (LaValle S.M, 2006), (Jaramillo-Botero et al., 2004). Geometrical trajectory generation and some techniques for its parameterization based on polynomial splines and function with trapezoidal velocity profile are an interest solution for this problem, actually, some of these methodologies are used for complex UAV trajectory definition nowadays. The novel solution presented in this book is the integration of these methods into a powerful environment that allows high-level user control of complex missions. The AVCL definitively brings those features and the next section will introduce some tests using the AVCL interpreter and the simulation environment.

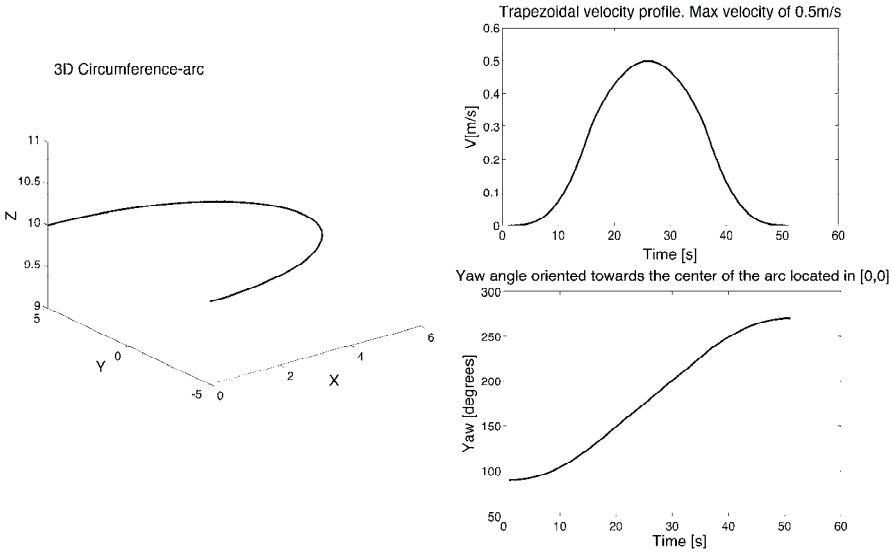

Figure 15. Circumference 3D motion, trapezoidal velocity profile and *yaw* angle evolution

4. TG²M Simulation Results

As shown in Fig. 1 the Mission Planner (MP) has two similar loops for mission planning and simulation/supervision. The difference is that in the Planning Loop the interpreter sends the projected waypoints back to the MP's Enhanced Reality, while in the Simulation Loop the interpreter commands the simulated vehicle, which in turn sends the simulated positions to

the MP. Our research group has developed a Simulink-based model of a UAV helicopter named *Vampira*, which includes a position controller and is capable of real-time simulation. This simulator has been used with the Mission Planning and Simulation tool to test the TG²M. For Mission Supervision the AVCL commands would be sent to the real vehicle, and its position plotted alongside the projected and/or simulated paths. The *Vampira* helicopter was built within the framework of the project: "Guidance and Control of an Unmanned Aerial Vehicle" DPI 2003 01767 (VAMPIRA), granted by the Spain Ministry of Science and Technology, and it will be used for the real-world tests of the built-in TG²M framework. Figure 16 shows the *Vampira* prototype, which includes: a GPS, Wi-Fi link, IMU, and a PC104 computer for the low-level control (main rotor and tail servos). The Vampira's dynamics model has been obtained, identified and validated in previous works (Valero, 2005), (del Cerro et al., 2004). This work takes advantage from the AVCL simulation capabilities in order to validate the TG²M framework theory for trajectory planning.

Figure 16. The *Vampira's* Helicopter prototype

Three test scenarios showcase the TG²M validation process. These tests involve the whole methodology previously presented in the other sections of this chapter, as well as the numerical simulation results using the AVCL environment and the embedded dynamics and control algorithms for the Vampira's helicopter. Two complex maneuvers are presented using 3D and 4D splines respectively and a simple last test using analytical function to generate a parameterized circumference motion.

1). *Semi-spiral using 3D splines for the velocity profile generation and the Frenet theory for UAV orientation:* In this first test, we used a 3D spline to joint three knot control points: (P_0(0, 0, 0), P_1(3, 5, 10), P_2(6, -7, 20)) at the desire time (given in seconds) for each point: (t(0, 10, 20)) and the desire initial and final speed (given in m/s): (V_0(0, 0, 0), V_f(0, -0.2, 0.4)):

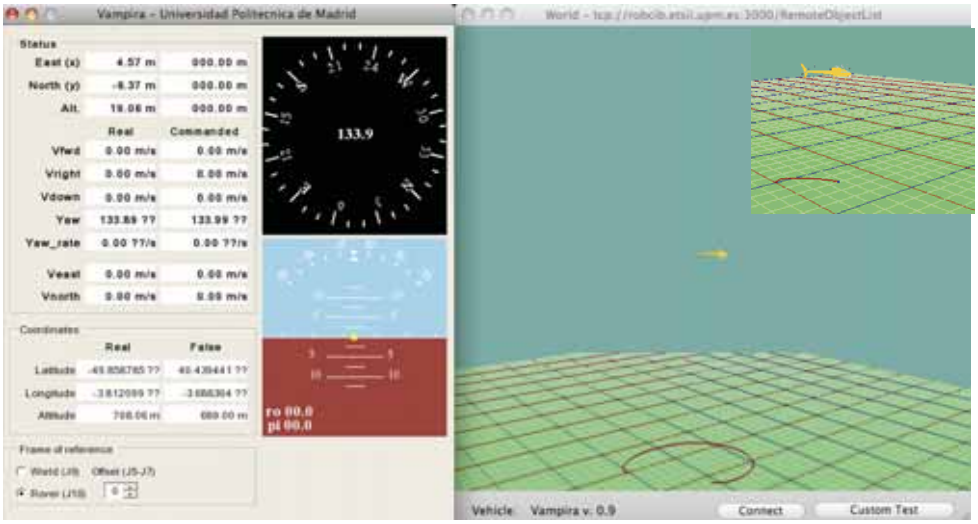

Figure 17. The AVCL simulation environment: Vampira's helicopter executing a semi-spiral motion using 3D splines

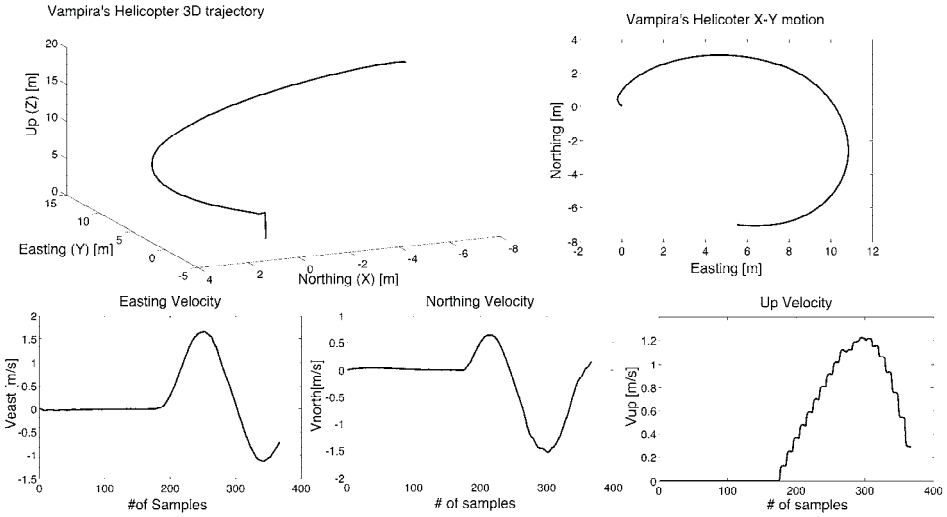

Figure 18. Test1: Cartesian UAV position and velocities with respect to the Inertial Frame

The UAV started from initial point located at (0, 0, 0) coordinate and finished its trajectory at (6, -7, 20). Visual simulation depicted in Fig 17, showed smooth motion across the trajectory due to the 3D spline approach. Nonetheless, 3D splines just allow the user to define the initial and final velocities of the motion, lacking of velocity control for the rest of the knot-points.

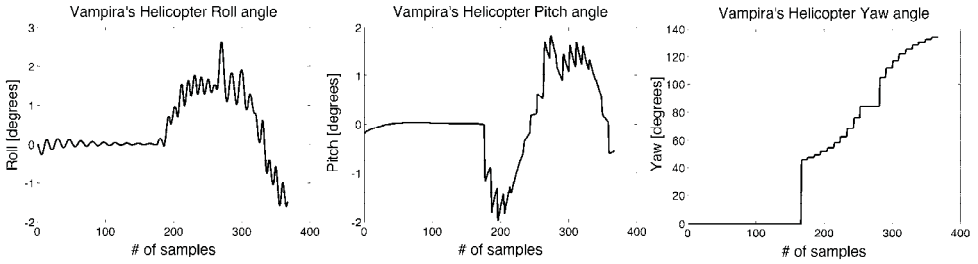

Figure 19. Test1: UAV orientation (Euler angles evolution)

To solve this problem, the following test introduces a more complex trajectory generation using 4D splines, addressing total user control of the UAV velocity profile.

2*). Complex trajectory using 4D splines for the velocity profile generation and the Frenet theory for UAV orientation*. This trajectory includes different kind of maneuvers joined into a single polynomial function (take-off, circumference-type motion and slow down in spiral-type motion). This test includes UAV long-endurance to high altitude (150 meters above ground) and a maximum easting displacement about of 60 meters. The following knot-control points (given in meters) have been defined: (P_0(0, 0, 0), P_1(0, 0, 20), P_2(0, 0, 40), P_3(0, 0, 60), P_4(10, 2, 80) , P_5(20, 4, 110) , P_6(25, -7, 130) , P_7(30, -10, 150), P_8(35, -5, 140) , P_9(30, 16, 125) , P_{10}(20, 5, 130) , P_{11}(33, -10, 145) , P_{12}(40, -5, 135) , P_{13}(55, -6, 125)):

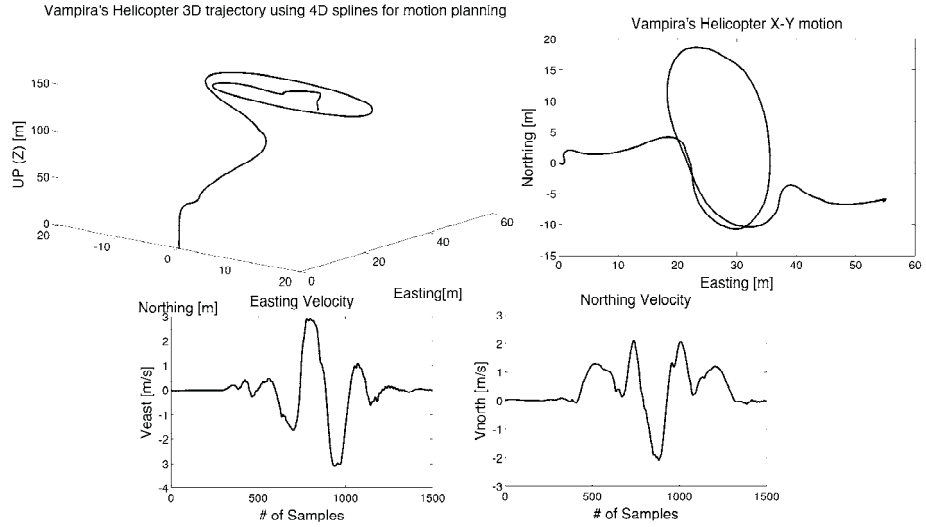

Figure 20. Test2: smooth 4D spline for complex maneuve

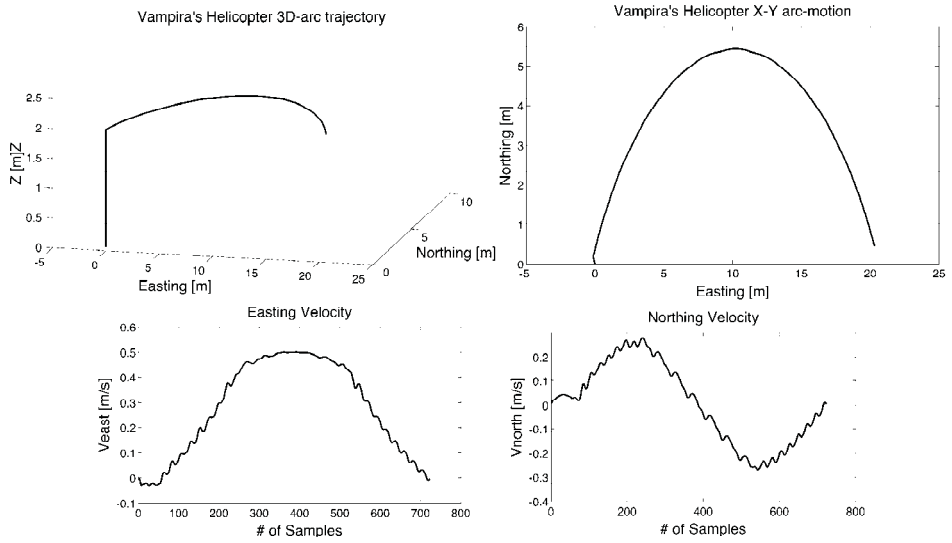

Figure 21. Test3: Arc-type motion using analytical functions

The advantage about using 4D splines relies in the possibility of defining feasible paths that matches with the knot-control points defined (with less match error percentage than the 3D polynomial splines). In addition, the user is able to define the set of velocities for each of the knot-points during the motion. The set of velocities (given in m/s) are: (V_0(0, 0, 0), V_1(0, 0, 0.8), V_2(0, 0, 1), V_3(0, 0, 1.2), V_4(0.5, 1, 1.4) , V_5(0, 0.5, 1.7) , V_6(2, 1.5, 2.5), V_7(3, 2.2, 3), V_8(2, 1.2, 2) , V_9(1, 0.5, 1.5) , V_{10}(-0.5, -1, 0.8) , V_{11}(-3, -2, 0.4) , V_{12}(-1, -0.5, 0.8), V_{13}(0, 0, 0)).

3). *Simple arc-type maneuver with trapezoidal velocity profile parameterization:* for the analytical AVCL feature of trajectory planning, the TG²M module supports straight-lines and circumferences motions. An arc defined by: P_0(0, 0, 2), P_1(10, 5.5, 2), P_2(20, 0, 2)) with a maximum velocity of 0.5m/s is tested using the AVCL interpreter that allows the user to define the trapezoidal velocity profile configuration. For this case, the acceleration slopes of the curves (see Fig. 21) have been set to the 30% of the total motion.

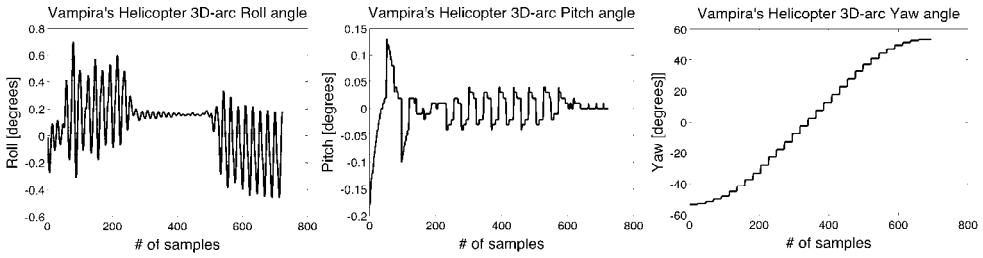

Figure 22. Test3: UAV orientation (Euler angles evolution)

5. Final Observations

For modeling continuous cartesian trajectories in the AVCL, several analytical functions and polynomial interpolation methods are available; all of which can be used in any combination. The TG²M module handles the definition of trajectories using knot control points as well as the incorporation of path constraints. It also supports the definition of complex tasks that require the construction of trajectories made up of different primitive paths in any combination of analytical and interpolated functions. The user-designed spatial trajectories can be visualized in three dimensions on the display window or plotted versus time using the embedded plotting library.

Simulation results have shown that the TG²M module works perfectly for the definition and testing of wide kind of smooth trajectories, allowing the user a high-level control of the mission due to the AVCL interpreter. The three different scenarios used for testing, allowed verifying that the mathematical framework used for the trajectory generation and guidance was really working during simulation flight. Percentage errors during maneuver execution were minimal, maintaining the UAV at the desired velocity limits and within the established path. We also incorporated velocity error fixing during flight. For high altitude tests, the velocity of the wind plays a mandatory role as a main disturbance external force. The TG²M module includes wind perturbation compensation. The Guidance module fixes the velocity commands in real-time flight maneuver, decreasing the error position tracking. For the three scenarios tests, the AVCL simulation environment includes normal wind conditions during simulation flight, introducing small perturbations into the UAV equations of motion. As shown in the obtained results, those perturbations were compensated, allowing the UAV to follow the desired trajectory within the less error as possible.

The Frenet-Serret formulas included for the UAV orientation also presented a good approach in order to obtain smooth UAV rotation rate during flight. The use of simple trigonometric theory to obtain and define the UAV orientation profile (*Yaw angle*) is not convenient for complex maneuvers. Splines sometimes require a lot of know-points for feasible trajectory guidance, hence, using these polynomial equations, the Frenet approach allowed smooth angle changes between knot-points, which it had not been obtained with the simple trigonometric angle calculation.

6. References

JAA & Eurocontrol,.A concept for the European Regulations for Civil Unmanned Aerial Vehicles. *UAV Task-Force Final Report*. 2004

Coifman, B., McCord, M., Mishalani, M., Redmill, K., Surface Transportation Surveillance from Unmanned Aerial Vehicles. *Proc. of the 83rd Annual Meeting of the Transportation Research Board*, 2004.

Held, Jason M; Brown, Shaun and Sukkarieh, Salah. Systems for Planetary Exploration. *I5th NSSA Australian Space Science Conference*, pp. 212-222, ISBN: 0864593740. RMIT University, Melbourne, Australia, from 14 to 16 September 2005.

Gutiérrez, P., Barrientos, A., del Cerro, J., San Martin., R. Mission Planning and Simulation of Unmanned Aerial Vehicles with a GIS-based Framework; *AIAA Guidance*, Navigation and Control Conference and Exhibit. Denver, EEUU, 2006.

Rysdyk, R. UAV path following for constant line-of-sight. In 2th AIAA Unmanned Unlimited. *Conf. and Workshop and Exhibit*, San Diego, CA, 2003.

Price, I.C, Evolving self organizing behavior for homogeneous and heterogeneous swarms of UAVs and UCAVS. *PhD Thesis*, Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2006.

Herwitz, S. Developing Military COEs UAV applications. *UAV Applications Center – NASA Ames Research Center*, Moffett Field, CA. 2007.

Alison A. P., Bryan G., Suresh K. K., Adrian A. K., Henrik B. C. and Eric N. J. Ongoing Development of an Autonomous Aerial Reconnaissance System at Georgia Tech . *UAV Laboratory*, School of Aerospace Engineering. Georgia Institute of Technology, 2003

Jaramillo-Botero A., Correa J.F., and Osorio I.J., Trajectory planning in ROBOMOSP, *Robotics and Automation Group* GAR, Univ. Javeriana, Cali, Colombia, Tech. Rep. GAR-TR-10-2004, 2004.

LaValle, S.M.: *Planning Algorithms*. Cambridge University Press (2006)

Moitra, A., Mattheyses, R.M., Hoebel, L.J., Szczerba, R.J., Yamrom, B.: Multivehicle 5reconnaissance route and sensor planning. *IEEE Transactions on Aerospace and Electronic Systems*, 37 (2003).

Zheng, C., Li, L., Xu, F., Sun, F., Ding, M.: Evolutionary Route Planner for Unmanned Air Vehicles. *IEEE Transactions on Robotics* 21 (2005) 609–620

Frazzoli, E. Maneuver-based motion planning and coordination for single and multiple UAVs. *AIAA's 1st technical conference and workshop on unmanned aerospace vehicles*. University of Illinois at Urbana-Champaign, il 61801. S. Portsmouth, Virginia. May 2002.

Angeles J., Fundamentals of Robotic Mechanical Systems. *Theory, Methods, and Algorithms*. Springer, New York, 1997.

Siouris, G. M. Missile Guidance and Control Systems. Springer, New York, 2004.

Valero, Julio. Modelo dinámico y sistema de supervisión de vuelo de un helicóptero autónomo. *Proyecto de fin de carrera*. ETSIIM-UPM. 2005.